A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging

Keunwoo Choi, György Fazekas, Mark Sandler
Centre for Digital Music, EECS
Queen Mary University of London
London, UK
keunwoo.choi@qmul.ac.uk

Kyunghyun Cho
Center for Data Science
New York University
New York, USA
kyunghyun.cho@nyu.edu

***Abstract*—In this paper, we empirically investigate the effect of audio preprocessing on music tagging with deep neural networks. While it is important to choose the best preprocessing strategy from an engineering perspective, it usually has been out of the focus in many academic research. We perform comprehensive experiments involving audio preprocessing using different time-frequency representations, logarithmic magnitude compression, frequency weighting, and scaling. We show that many commonly used input audio preprocessing techniques are redundant except logarithmic magnitude compression.**

***Index Terms*—deep learning, music tagging, convnet, audio**

I. Introduction

Music information retrieval researches that use deep learning techniques commonly focus on optimising the hyperparameters which specify the network structure. Conversely, the audio preprocessing stage is often decided on using heuristics without being subject to optimisation.

Although neural networks are known to be universal function approximators [1], training efficiency and performance may vary significantly with not only different training methods but also generic techniques including preprocessing the input data [2]. In other words, a neural network can *represent* any function but it does not mean it can always *learn* any function in practice. Therefore, both empirical decisions and domain knowledge are crucial in applying deep learning. Choosing between various preprocessing methods can be seen as a non-differentiable choice function, which cannot be optimised using gradient-based learning methods.

To consider examples in prior works, melspectrograms have been preferred over short-time Fourier transform in many tasks [3] because it was considered to represent enough information about many problems despite of its smaller size. As another example, when a time-frequency representation magnitude $\mathbf{X} \in \mathbb{R}_{\geq 0}^{N \times M}$ is given, one of the most common preprocessing approaches is to apply logarithmic compression, i.e., $\log(\mathbf{X}+\alpha)$ where α can be arbitrary constants such as very small number (e.g. 10^{-7}) or 1. However, the performances of these methods are not usually compared.

In this paper, we focus on audio preprocessing strategies for deep convolutional neural networks for music tagging. We choose music tagging task because *i)* it has the largest public dataset which enables large-scale deep learning experiments and *ii)* the diversity in music tags, e.g., genre, mood, instrument, era, implies that the analysis would generalise to other music tasks such as genre classification. By assessing various preprocessing strategies and providing empirical results, we aim at demystifying the effects of audio preprocessing on network performance. This will help researchers in designing deep learning systems for music research.

II. Experiments and Discussions

To compare the effects of audio preprocessing, a representative network structure needs to be defined first. A ConvNet (convolutional neural network) with 2D kernels and 2D convolution axes was chosen. This showed good performance with efficient training in a prior benchmark study [4], where the model we selected was denoted *k2c2*, indicating 2D kernels and convolution axes. As illustrated in Figure 1, homogeneous 2D (3×3) convolutional kernels are used in every convolutional layer. The input has a single channel, 96 mel bins, and 1,360 temporal frames, denoted (1, 96, 1360). Exponential linear unit (ELU) is used as an activation function in all convolutional layers [5].

To train our music tagger, we used the Million Song Dataset (MSD) [6] with preview audio clips. The training data are 30-second stereo mp3 files with a sampling rate of 22,050Hz and 64 kbps constant bit-rate encoding. For efficient training in our experiments, we downmix and downsample the signals to 12 kHz after decoding and trim the audio duration to 29-second to ensure equal-sized input signals. The short-time Fourier transform and melspectrogram are computed using a hop size of 256 samples (21.3 ms) with a 512-point discrete Fourier transform aggregated to yield

FAST IMPACt EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons (688382). Mark Sandler acknowledges the support of the Royal Society as a recipient of a Wolfson Research Merit Award. Kyunghyun Cho thanks the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016)

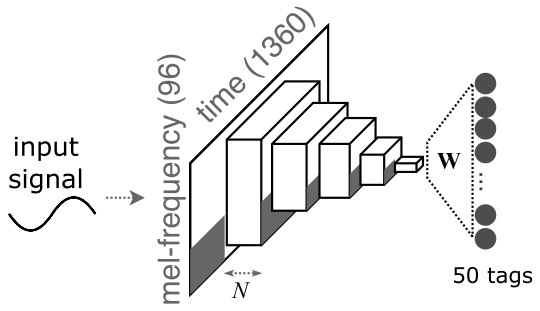

Fig. 1. Network structure of the 5-layer ConvNet. N refers to the number of feature maps (which is set to 32 for all layers in this paper) while **W** refers to the weights matrix of the fully-connected output layer.). Max-pooling is applied after each convolutional layer with sizes of (2, 4), (4, 4), (4, 5), (2, 4), (4, 4) respectively (along time and frequency axes).

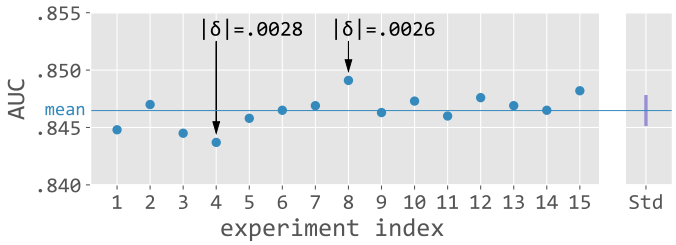

Fig. 2. AUC scores (left), their mean, and the standard deviation of the AUC scores (right). The two deltas on the plot indicate the difference between the average AUC and the scores of experiments 4 and 8.

96 mel bins per frame. The preprocessing is performed using *Librosa* [7] and *Kapre* [8]. A total of 224,242 tracks are used and split into train/validation/test sets comprising 201,672/12,633/28,537 tracks respectively.[1] During training, the binary cross-entropy function is used as a loss function. For the acceleration of stochastic gradient descent, we use adaptive optimisation based on Adam [9]. The experiment is implemented in Python with *Keras* [10] and *Theano* [11] as deep learning frameworks.

In the all experiments, area under curve - of receiver operating characteristic (AUC) is used as a metric. Although it can be lower than 0.5 in theory, AUC practically ranges in [0.5, 1.0] since random and perfect predictions show an AUC of 0.5 and 1.0 respectively. The reported AUC scores are all measured on the test set only.

A. Variance of network initialisations

In deep learning, using K-fold cross-validation is not a standard practice for two reasons. First, with large enough data and a good split of train, validation, and test sets, the model can be trained with small variance. Second, the cost of hyperparameter search is very high and it makes repeating experiments too expensive in practice. For these reasons, we do not cross-validate the ConvNet in this study. Instead, we present the results of repeated experiments with fixed network and training hyperparameters, such as training example sequences and batch size. This experiment therefore measures the variance of the model introduced by different weight initialisations of the convolutional layers. For this, a normal distribution is used following He et al. [12], which has been shown to yield a stable training procedure.

The results are summarised in Figure 2. This shows the AUC scores of 15 repeated experiments on the left as well as their standard deviation on the right. Small standard deviation indicates that we can obtain a reliable, precise score by repeating the same experiments for a sufficient number of times. The two largest differences observed between the average AUC score and that of experiment 4 and 8 (AUC differences of 0.0028 and 0.0026 respectively) indicate that we may obtain up to ~ 0.005 AUC difference among experiment instances. Based on this, we assume that an AUC difference of < 0.005 is non-significant in this paper.

[1]The network implementation and split settings are published online:
`https://github.com/keunwoochoi/transfer_learning_music`
and
`https://github.com/keunwoochoi/MSD_split_for_tagging`

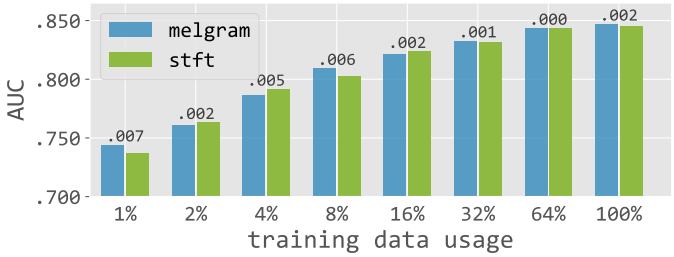

Fig. 3. Performances of predictions with melspectrogram and STFT with varying training data sizes. The numbers above bars indicate the absolute performance differences between melspectrograms and STFTs.

B. Time-frequency representations

STFT and melspectrogram have been the most popular input representations for music classification [3]. Although sample-based deep learning methods have been introduced, 2-dimensional representations would be still useful in the near future for efficient training. Melspectrograms provide an efficient and perceptually relevant representation compared to STFT [13] and have been shown to perform well in various tasks [14]–[18]. However, an STFT is closer to the original signal and neural networks may be able to learn a representation that is more optimal to the given task than melspectrograms. This requires large amounts of training data however, as reported in [18] where using melspectrograms outperformed STFT with a smaller dataset.

Figure 3 shows the AUC scores obtained using $\log$(STFT) vs. $\log$(melspectrogram) while varying the size of the utilised training data. Although there are small differences on AUC scores up to 0.007, neither of them outperforms the other, especially when enough data is provided. This rebuts the results reported in [18] because melspectrograms did not have a clear advantage here, even with a small training data size. This may be explained by the higher frequency resolution of the STFT representation used and summarised as follows.

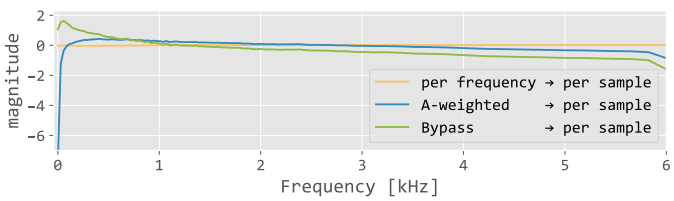

Fig. 4. Average frequency magnitude of randomly selected 200 excerpts with three frequency-axis normalisation. A per-sample (excerpt) standardisation follows to remove the effect of different overall average magnitude.

- STFT in [18]: $6000/129{=}46.5$ Hz (256-point FFT with 12 kHz sampling rate)
- STFT in our work: $6000/257{=}23.3$ Hz (512-point FFT with 12 kHz sampling rate)
- Melspectrogram in [18] and our work: 35.9 Hz for frequency $<$ 1 kHz (96 mel-bins, implementation by [19])

In [18], the frequency resolution of the STFT was lower than that of the melspectrogram to enable comparing them with similar number of frequency bins. On the contrary, STFT of higher frequency resolution is used in our experiment. Nevertheless, it is found to be only as good as melspectrogram in terms of performance. This means overall that, in practice, melspectrogram may be preferred since its smaller size leads to reduced computation in training and prediction. However, this may be because the limit of the used networks which does not take advantage of finer input rather than because of the nature of the task. For example, detecting vocal component (to tag 'instrumental') could use finer frequency structure of consonant part, but only if the network is designed to capture them effectively. The figure also illustrates how much the data size affects the performance. Exponentially increasing data size merely results in a linear AUC improvement. AUC starts to converge at `64%` and `100%`.

C. Analysis of frequency-axis weights and scaling effects

In this section, we discuss the effects of magnitude manipulation. Preliminary experiments suggested that there might be two independent aspects to investigate; *i)* frequency-axis weights and *ii)* magnitude scaling of each item in the training set. Examples of frequency-axis weights are illustrated in Figure 4, where different weighting schemes are plotted. Our experiment is designed to isolate these two effects. We tested two input representations *log-melspectrogram* (melspectrogram in decibel scale) vs. *melspectrogram*, with three frequency weighting schemes *per-frequenc y*, *A-weighting* and *bypass*, as well as two scaling methods $\times 10$ (`on`) and $\times 1$ (`off`), yielding $2{\times}3{\times}2{=}12$ configurations in total.

We summarise the mechanism of each block as follows. First, there are three frequency weights schemes.

- `per-frequency stdd`: This is often called spectral whitening. We compute means and standard deviations across time, i.e., per-frequency and standardise each frequency band using these values. The average frequency response becomes flat (equalised). This method has been used in the literature for tagging [18], singing voice detection [20] and transcription [21].
- `A-weighted`: We apply the international standard IEC 61672:2003 A-weighting curve, which approximates human perception of loudness as a function of frequency.
- `Bypass`: Do not apply any processing, i.e., $f : \mathbf{X} \to \mathbf{X}$

There are two other blocks which are mutually independent and also independent to frequency weights schemes.

- `per-sample stdd`: Excerpt-wise normalisation with its overall mean and standard deviation, i.e., using statistics across time and frequency of each spectrogram.
- `×10 scaler`: Multiply the input spectrogram by 10, i.e.,$f : \mathbf{X} \to 10\mathbf{X}$.

In the following sections, we discuss result on frequency weighting and scaling effects respectively.

1) Frequency weighting: This process is related to loudness, i.e., human perception of sound energy [13], which is a function of frequency. The sensitivity of the human auditory system drops substantially below a few hundred Hz[2], hence music signals are often produced to exhibit higher physical energy in the lower frequency range in order to be perceptually balanced. This is illustrated in Figure 4, where uncompensated average energy measurements corresponding to the *Bypass* curve (shown in green) yield a peak at low frequencies. This imbalance affects neural network activations in the first layer which may influence performance. To assess this effect, we tested three frequency weighting approaches. Their typical profiles are shown in Figure 4. In all three strategies, excerpt-wise standardisation is used to alleviate scaling effects (see Section II-C2).

Our test results show that networks using the three strategies all achieve similar AUC scores. The results are illustrated in Figure 5 where the colours indicate normalization schemes.[3] The performance differences *within* four groups {`1`, `1s`, `2`, `2s`} shown in Figure 5 are small and none of them are significantly different the others, which indicates they do not produce a significant effect on the training. The curves in Figure 4 show the average input magnitudes over frequency. These offsets change along frequency, but the change does not seem large enough to corrupt the local patterns due to the locality of ConvNets, and therefore the network is learning useful representations without significant performance differences within each group.

2) Analysis of scaling effects: For a number of reasons discussed below, we may assume a performance increase if we scale the overall input magnitudes. During training using gradient descent, the gradient of error with respect to weights $\frac{\partial E}{\partial W}$ is proportional to $\frac{\partial}{\partial W} f(W^\top X)$ where f is the activation function. This means that the learning rate of a layer is proportional to the magnitude of input X. In particular, the first layer usually receives the weakest error backpropagation, hence scaling of the input may affect the overall performance.

[2]See equal-loudness contours e.g. in ISO 226:2003.

[3]Additionally, {`1`, `2` vs. `1s`, `2s`} compares scaling effect (see Section II-C2) and {`1`, `1s`} vs. {`2`, `2s`} compares the log-compression effect (see Section II-D).

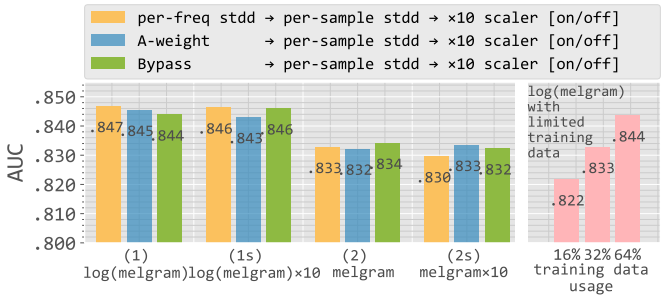

Fig. 5. Performance comparisons of different preprocessing procedures. From left to right, four groups of scores are from different magnitude processing (melspectrogram in decibel scale and linear scale), with additional $\times 10$ scaler turned on/off. In each group, yellow/blue/green bars indicates different frequency-axis processing as annotated in the legend. Logarithmic compression is done before other preprocessings are applied.

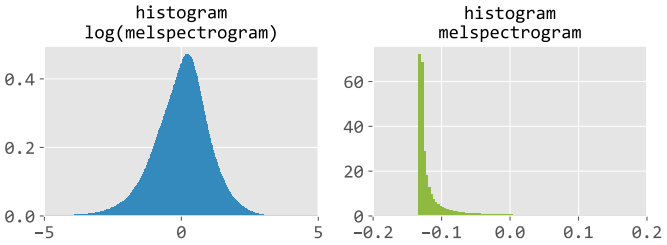

Fig. 6. histograms of the magnitude of melspectrogram time-frequency bins with (left) and without (right) logarithmic compression. The number of bins are 100 and both are normalised, i.e., $\sum_{i=1}^{100} 0.01 \times y_i = 1$. Log compression significantly affects the histogram, making the distribution Gaussian (left), otherwise extremely skewed (right). This is after standardisation and based on randomly selected 100 tracks from the training set.

We tested the effect of this with the results shown in Figure 5. To this end, consider comparing the matching colour bars of {1 vs. 1s} and {2 vs. 2s}. Here, the scaling factor is set to 10 for illustration, however many possible values <100 were tested and showed similar results. In summary, this hypothesis is rebutted as scaling did not affect the performance. The analysis of trained weights revealed that different magnitudes of the input only affects the bias of the first convolutional layer. Training with scaling set to $\times 10$ results in about 3.4 times larger mean absolute value of the biases in the first layer. This may be due to batch normalization [22] which compensates for the different magnitudes by normalising the activations of convolutional layers.

D. Log-compression of magnitudes

Lastly, we discuss how logarithmic compression of magnitudes, i.e. decibel scaling, affects performance. This is considered standard preprocessing in music information retrieval. The procedure is motivated by the human perception of loudness [13] which has logarithmic relationship with the physical energy of sound. Although learning a logarithmic function may be a trivial task for neural networks, it can be difficult to implicitly learn an optimal nonlinear compression when it is embedded in a complicated task. For the same reason, a nonlinear compression was also shown to affect the performance in visual image recognition using neural networks [23]. Figure 6 compares the histograms of the magnitudes of time-frequency bins after zero-mean unit-variance standardisation. On the left, a logarithmically compressed melspectrogram shows an approximately Gaussian distribution without any extreme values. Meanwhile, the bins of linear melspectrogram on the right is extremely condensed in a very small range while they range in wider region overall. This means the network should be trained with higher numerical precision to the input, hence more vulnerable to noise.

As a result, log-compressed melspectrograms always outperform the linear versions as shown in Figure 5, where matching colour bars should be compared across within {1 vs. 2} and {1s vs. 2s}[4]. The additional work introduced by *not* using log-compression can be roughly estimated by comparing these scores to those networks when the training data size is limited (shown in pink on the right of Figure 5). While this also depends on other configurations of the task, seemingly twice the data is required to compensate for the disadvantage of not using a log-compressed representation. This is a significant difference as the non-trivial burden for collecting labelled data and training time. It is noteworthy that this specific 'doubling' relationship would differ by the task and data, e.g. [24].

III. Conclusion

In this paper, we have shown that some of the input preprocessing methods can affect the performance of neural networks for music tagging. We quantify this in terms of the size of the training data required to achieve similar performances. Among several preprocessing techniques tested in this study, only logarithmic scaling of the magnitude resulted in significant improvement. In other words, the network was resilient to most modifications of the input data except logarithmic compression of magnitudes in various time-frequency representations. Considering the diverse nature of music tags that covers genre, mood, and instruments, our results provide practical advice applicable in many similar machine-listening problems, e.g., music genre classification or mood prediction.

References

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[2] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[3] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "A tutorial on deep learning for music information retrieval," *arXiv preprint arXiv:1709.04396*, 2017.

[4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units," *arXiv:1511.07289*, 2015.

[6] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*. University of Miami, 2011, pp. 591–596.

[4]Log-compressed STFT also outperformed linear STFT in our initial experiments.

[7] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.293021

[8] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*, 2017.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." 3rd International Conference for Learning Representations, San Diego, 2015, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[10] F. Chollet, "Keras: Deep learning library for theano and tensorflow," https://github.com/fchollet/keras, 2015.

[11] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590*, 2012.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[13] B. C. Moore, *An introduction to the psychology of hearing*. Brill, 2012.

[14] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2643–2651.

[15] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[16] J. Schluter and S. Bock, "Improved musical onset detection with convolutional neural networks," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*. IEEE, 2014.

[17] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014), Taipei, Taiwan*, 2014.

[18] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *The 17th International Society of Music Information Retrieval Conference, New York, USA*. ISMIR, 2016.

[19] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, p. 1998, 1998.

[20] J. Schlüter, "Learning to pinpoint singing voice from weakly labeled examples," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 44–50.

[21] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic music transcription," *arXiv preprint arXiv:1508.01774*, 2015.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] S. F. Dodge and L. J. Karam, "Understanding how image quality affects deep neural networks," *CoRR*, vol. abs/1604.04004, 2016. [Online]. Available: http://arxiv.org/abs/1604.04004

[24] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," *arXiv preprint arXiv:1711.02520*, 2017.